Spectral Code Index (SPECOIND): A General Infrared Spectral Database Search Method

JIAN FENG LI, BO TAO FAN,* JEAN-PIERRE DOUCET, and ANNICK PANAYE

Interfaces, Traitements, Organisation et Dynamique des Systèmes, CNRS UMR 7086, Université Paris 7, 1, rue Guy de la Brosse, 75005 Paris, France

A new spectral code that can be used by Relational Database Management Systems (RDBMS) as an index for infrared (IR) spectra searches in Relational Database (RDB) is presented and its suitability is evaluated. Spectral codes are constructed for all spectra in the database as the spectral indexes and three query strings are created with the same theory used for the creation of the index code for the query spectrum. Some effects of parameters used to create index strings and query strings are discussed. All spectral searches are accomplished in structured query language (SQL) approach and the utilization examples of SQL have been shown. The sequential application of this procedure can reduce the original library of about 18 000 spectra to a few spectra that can be used as references for subsequent detailed comparison. The software developed for the proposed system is particularly suitable for spectral search and structure interpretation.

Index Headings: **Infrared spectroscopy**; **Spectral database**; **Spectral code index; Index**; **Code string.**

INTRODUCTION

Because of the limitations of the knowledge of the natural world or the calculation power, empirical research methods, not calculation, play a significant role in the chemical research field. The database is the data infrastructure of any empirical research. The spectra are a very important resource in chemistry applications and research domains. Searching spectral libraries of known compounds is one of the most important tools in spectral interpretation and structure elucidation.[1–4] The key to using this tool is that the spectral collection must be large enough to give a decent probability of a match. In 1991, Warr[5] noted that ''no single spectral database contains more than approximately 100 000 compounds even though there were ten million known chemical compounds in the CA Registry System. There were 20–30 times as many hard-copy libraries of spectra as there were computer-readable spectral libraries''. The situation is changing. For example, Bio-Rad's Informatics Division[6] announced that it had the largest collection of infrared (IR) data in the world of over 220 000 spectra of pure organic and commercial compounds. Before the 1990s, most of the spectra were stored in printed format,[7–12] and some of the handbooks have indexes on compound name, molecular formula, chemical class, numerical, and SpecFinder Code. Nowadays, there are more and more digital and on-line spectral databases that can provide more comprehensive search tools and higher quality spectra. FTIRSearch[13] and Chemical Concepts[14] are big-volume commercial databases, and databases like SpecInfo,[15] NIST Chemistry WebBook,[16] and Integrated Spectral Data Base System for Organic Compounds (SDBS)[17] are famous on-line free-access public databases. Because spectral databases are very useful, not only big companies but also many research groups have given attention to creating and managing databases. As the power of personal computers has grown stronger, some publications can be found that discuss the application of personal computers to creating an IR spectral database and a spectral retrieval system for a local group.[18–23]

Reliability, ease-of-use, and ease-of-management are important factors, especially to large-volume databases. As scientists began to rely more heavily on computerized research data, it became increasingly clear that the traditional file-based methods of storing and retrieving data were both inflexible and cumbersome to maintain. Because application code for accessing the data contained hard-coded pointers to the underlying data structures, a new data analysis could take months to start. Even minor changes were complicated and expensive to implement. Otherwise, these self-defined file systems almost always lack well-defined technical standards and stable technical support. These real research needs drove the introduction of the relational database into the scientific arena.

Applications that are well suited to the tabular representation of data, with a low level of relationship complexity, are best served through relational technology, since at the very highest level, Relational Database Management Systems (RDBMSs) manipulate rows and columns of data, using multiple indexing and joins to associate tables. The true power of a relational database (RDB) resides in its ability to break the link between data access and the underlying data itself. RDBMSs offer a wide variety of functionality such as compound properties storage, display, portability, scalability, and robustness. They can store and manage all data types that a chemist encounters in the real world. Unlike a strictly hierarchical system, one of the great strengths of an RDBMS is its ability to accept high-level dynamic queries from users who have no knowledge of the underlying data structures. Using a high-level access language, Structured Query Language (SQL), one can access and manage all corporate data dynamically without any knowledge of how the underlying data is actually stored. SQL technology is easy to use and highly efficient. We can find some examples of using standard commercial database systems like Oracle or Access to manage spectral databases or other chemical information.[24,25]

Archiving full spectra rather than only stronger peaks

Received 9 September 2002; accepted 10 February 2003.
* Author to whom correspondence should be sent.

0003-7028/03/5707-0858$2.00/0

is popular in laboratories because a powerful computer is easily available. The keys to full spectral searching in huge-volume databases are making the process fast enough and getting real, relevant search results. Because every data point in every spectrum in the database must be compared to the corresponding data points in the query spectrum, for bigger databases, mechanical retrieval and comparison procedures require a lot of calculations and are tiresome. Although the entire database can be searched, it is clearly faster to limit the number of spectra to be compared by some selection rules. To improve the search speed, researchers may choose different approaches. One of these is to reduce the resolution of the spectra to be archived in the spectral domain.[26] A one-fold decrease in the resolution of a spectrum leads to one half of the computational time and space needed. The second method to speed up data processing is to compress the spectra to produce a smaller data set. The most commonly used compression techniques include the Fourier transform,[27,28] wavelet transform or its variants,[29,30] principal component analysis,[31–33] etc. Another commonly used method is implemented by creating a special file that includes the most important properties of a spectrum, e.g., stronger peaks and their locations for peak search.[34–37] All these methods create some kind of index that has a different management method from normal RDBMS and that can not be compared directly by SQL. Because these indexes cannot be sorted, no sorting algorithm from RDBMS can be used to reduce comparisons. The aim of this kind of index is to reduce the pre-search volume. When a spectrum is submitted to the database with this kind of index, the system must compare all the index properties one by one of the unknown spectrum with each reference spectrum in the index file before launching a detailed comparison. That is to say, every search must scan the all spectral indexes.

An index in an RDB is a list of sorted table values with the storage locations of rows in the table that contain each value. An index allows the database program to find data in a table without scanning the entire table. Generally, searches for rows that match a specific search key value (an exact match query), for rows with search key values in a range of values (a range query), or for rows that match a specific pattern of search key values will benefit from using indexes.

In this paper, a method to construct a spectral database in an RDBMS, which has obvious advantages for management and searching, is introduced. For this large-volume spectral database, a SPEctral CODe INDex (SPECODIND) system, which enables users to make use of RDBMS to search spectra using SQL, will be presented and parameters for the creation of this index system will be discussed.

METHODS

Coding Scheme. In RDBMS, a row with different fields represents a record and some fields should be key properties that can be used as search keys. A spectrum can be represented as a series of wavelengths and corresponding intensities; they can be saved in two columns or in one row that has as many fields as wavelength intervals in the table. The first spectral storage style saves one spectrum in a number of rows and there is no field that can be considered as a key property for a spectral search because a spectrum is dispersed in many rows. For the second storage type, even though a spectrum is stored in one row as a normal database record, no field has meaning to the IR spectral search because the IR spectral search cannot be accomplished by searching a single point on the spectrum, but rather, a group of points on the spectrum must be searched. For these reasons, it is impossible to search spectra directly by SQL from these two styles of data storage.

Identifying an appropriate set of indexes for a database system is a complex undertaking. At the same time, there are some limits to pattern searching, which is an important fuzzy search method: only a character string or a data type that can be converted to a character string can use the ''LIKE'' key word in SQL pattern matching in RDBMS. As a result, if spectra can be processed in string format they will be searchable by the flexible query methods in SQL. Consequently, in the system presented here, every spectrum in the library will be assigned a corresponding code string and stored in a separate table, called an index table, which can be used directly for SQL searching.

A spectrum, appearing on the spectrogram as bands, can be described in terms of variables: position, intensity, etc. Any band character can be used to create an index in a suitable form. Absorption bands may represent predominantly a single vibrational mode. Certain absorption bands, for example, those arising from C–H, O–H, and C=O stretching modes, remain within fairly narrow regions of the spectrum. The intensity can vary dramatically since the dipole moment change depends entirely upon what is attached to each carbon. The intensity also changes with the weight or the concentration of samples, etc. That is to say, the intensity information is less important than the locations of bands and it is not the characteristic property. Hence, it is not helpful to include intensity information in indexes. The binary spectral representation also omits the intensity and it can be used as an index as well;[38] but it is too coarse for highly efficient searching.

The main steps in converting a spectrum to a code string are dividing sections and mapping peaks. Dividing sections is to divide a spectrum into finite sections, and mapping peaks is to map the peak positions of a spectrum in divided sections onto letters. Detailed steps to create code string are described below:

(1) Creating code sections: The spectrum to be coded is divided into sections by specific resolution increments with predefined widths. The more intervals, the higher the precision and the lower the comparison tolerance. The extreme instance is the number of intervals equal to the number of data points of a spectrum. Here, for convenience in later discussion, the number of the sections is defined as the first level of the coding scheme.
(2) Choosing the peak: Every peak should be mapped into a section. When there is more than one peak located in the same section, the following rules are used: if the intensities of peaks are different, choose the strongest peak; if candidates have the

same intensity, the peak with the smaller wavenumber is chosen.

(3) Coding: After the candidate peaks in each section are found, the location of each candidate peak is normalized from 0 to 1 in its own section. If there is no peak in the coding section, it is marked as ''No Peak'' and the section is assigned a ''0''. Otherwise, the peaks are converted to an alphabetic letter according to their normalized location values using the formulas below. One can use all printable letters in the ASCII table to represent the location of the peak in one section. Normally, one prefers to use 26 capital or small alphabetic letters that begin at ASCII65 or ASCII97, resulting in 26 choices. The number of letters used for coding is defined as the second level of the coding scheme. If two letters are used for coding in a section, the second level of this coding scheme is 2.

(4) Constructing the string: After all sections are processed, link all letters in each section sequentially to make up a whole code string.

The formula and its explication are listed below:

$$SectionWidth = (SpectrumEnd - SpectrumStart)/SectionNumber \quad (1)$$

$$MappedPeakPos = (SpectrumEnd - PeakPos)/SectionWidth \quad (2)$$

$$ArrayIndex = Integer(MappedPeakPos) \quad (3)$$

$$CODESTR(ArrayIndex) = Integer[(MappedPeakPos - ArrayIndex) \times Subprecision] \quad (4)$$

SectionNumber: the first level of the coding scheme; *SectionWidth*: the width of a coding section; *SpectrumStart, SpectrumEnd*: start and end wavenumber of the spectrum; *MappedPeakPos*: normalized peak location; *ArrayIndex*: the element number of the code string; *CODESTR*(): the code string; *Subprecision*: the second level of the coding scheme.

For example, for a spectrum ranging from 3997 to 447 cm^{-1}, *SpectrumEnd* = 3997 cm^{-1} and *SpectrumStart* = 447 cm^{-1}, respectively. If a peak is located at 3397.285 cm^{-1}, the parameters are: *PeakPos* = 3397.285 cm^{-1}, *SectionNumber* = 18, *Subprecision* = 2. Following Eqs. 1–4, we can get:

$$\begin{aligned} SectionWidth &= (3997 - 447)/18 \\ &= 197.222\ cm^{-1} \\ MappedPeakPos &= (3997 - 3397.285)/197.222 \\ &= 3.041 \\ ArrayIndex &= Integer(3.041) = 3 \\ CODESTR(ArrayIndex) &= Integer[(3.041 - 3) \times 2] \\ &= 0 \rightarrow ''A'' \end{aligned}$$

The final example code string is shown in Fig. 1.

Section	0	1	2	3	...	17
Code Letter	0	...	A	A	...	

FIG. 1. An example of a code string.

Selecting Wavenumber for Index. For every spectrum in the database, all peaks with intensities that are stronger than the predefined criterion are encoded and stored in the index table. Commonly, an IR spectrum is interpreted in two regions: the functional group region and the fingerprint region. Most researchers use 1500 cm^{-1} as the dividing point between these two regions; at the same time, one can find other personal preferences.[39,40] The functional group region provides evidence of functional groups in a molecule based on the absorbance frequency. Peaks in this region are characteristic of specific kinds of bonds and therefore can be used to identify whether a specific functional group is present. The peaks in the fingerprint region may be characteristic of molecular symmetry, or may be combination bands arising from multiple bonds deforming simultaneously. The importance of the fingerprint region is that each different compound produces a different pattern of troughs in this part of the spectrum. In this region, very small differences in structure can lead to differences in absorbances. Additionally the ''fingerprint'' is generally more complex and less specific, rendering identification of individual bonds more difficult than it is in the ''cleaner'' region at higher wavenumbers. For a similarity search, these peaks will preclude some candidates for rigorous searching conditions. It is obvious that any region of the spectrum or the whole region can be used to create the index table and the final choice depends on the application. Because an SQL spectral search is a pre-search method for spectral search and the similarity search is useful for spectrum interpretation and structure elucidation, the functional group region between 4000 and 1500 cm^{-1} was chosen to create the index table.

SQL Searching Method. As a high-performance query language, SQL provides enough comparison operators for powerful customizable searching. It can compare two database records exactly. At the same time, for a similarity search, one can use the ''LIKE'' key word in the query sentence to determine whether or not a given spectrum matches similar spectra in the index table. Since the spectra in the index table are character strings, a search pattern can include regular characters and wildcard characters. In pattern matching, regular characters must exactly match the characters specified in the character string. Wildcard characters, however, can be matched with arbitrary fragments of the character string. Using wildcard characters makes the ''LIKE'' operator more flexible than using the ''='' and ''!='' string comparison operators. Here are the wildcard characters used in SQL and their meanings:

%: Any string of zero or more characters.
_: Any single character.
[ ]: Any single character within the specified range ([a–f]) or set ([abcdef]).
[^]: Any single character not within the specified range ([^a–f]) or set ([^abcdef]).

The unknown spectrum to be searched is transformed

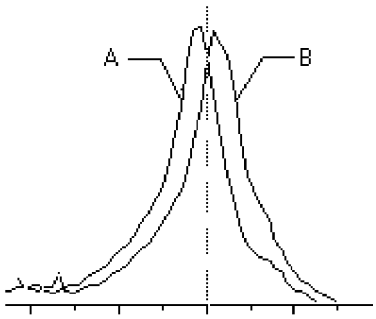


FIG. 2. Two overlapped bands. Most of the parts of the two bands are overlapped but their peak locations are in different sections. For a code system with a second code level of 4, the code for spectrum A will be ''D'' and for spectrum B will be ''A''. In general, these two bands should be considered as two matched bands. But in the general code steps, these two bands are different.

into a spectral code string using the same method used for the construction of the spectral index strings before the start of the search. To use SQL with the ''LIKE'' key word for searching spectra in a database, we should change some letters in the code string of a query spectrum to a spectral query pattern by including valid SQL wildcard characters. Here we define our special meanings for searching the user input spectrum:

0:	No peak.
!0:	Must have peak.
Alphabetic letters:	There is a peak in this section, and the letter represents the location.
%:	No matter what it is.

Preprocessing of the Query Spectrum. In our coding system, only the location of a band is processed. When we encode a spectrum, some detail information will definitely be lost. In order to compare the bands of the query spectrum and the spectra in the database, which have different band widths generically, the band widths of the query spectrum will be normalized before it is encoded. We define a width window to accomplish this aim. When a band is narrower than the width of the window, it is extended to the width of the widow. Otherwise, the band will not be processed in this step. When all bands have been normalized, a routine is used to check if any band is extended to regions of other bands. If two neighbors are overlapped, the overlapped bands are merged to become a new band. Only the location of the peak with the higher wavenumber is kept. The peak location is always encoded directly. If two separated bands are mapped into the same main section and have different code letters, a ''[!0]'' is marked for this section. The other sections occupied by the band are marked as ''?''. The width window allows the SQL to compare the bands in different widths. We name the query string created in this step the fuzzy query string.

Shifted String. Like the similar bands shown in Fig. 2, which mainly overlap, the locations of the peak maxima vary by one or more resolution increments because of variations between instruments and the reliance on published spectra.[35] These peaks may have different code letters when the peak positions are near the borders of sections because of the digitalization of a continuous spectrum into discrete peak locations. Theoretically, peaks with this feature in adjacent resolution increments in the target and reference spectra should not be counted as mismatches, and some algorithms should be applied to permit a variable ''wiggle'' or a tolerance range must be allowed in matching peak locations. Anderson and Covert[41] built in a ± 0.1 μm tolerance for each original coded spectral peak. The Sadtler Handbook of IR spectra allows a tolerance of ± 10 cm^{-1} when comparing the codes of the unknown against the Spec-Finder Codes.[42] Heite et al.[38] applied windows 0.3 and 0.5 μm wide on both sides of a peak position. Tanabe et al.[43] proposed a tolerance for the peak position data as a function of λ_{max}. Penski et al.[34] proposed a wavelength correction procedure to correct for linear wavelength displacement. Lau et al.[44] proposed a method that employs statistics to define the tolerance range for the variations in peak locations due to indeterminate errors in measurements instead of using the rather arbitrary methods of defining the tolerance range.

In this system, we applied a different method in concert with using SQL. To solve the above problem, two kinds of query strings—right and left shifted strings—are used with the standard query string. To construct a shifted string, the query spectrum is shifted to the high or low end of the wavenumber in 0.4 of the length of the subsection that is defined by the second-level coding scheme. This value is selected randomly. When the query spectrum has been moved a small distance, it is encoded using the standard coding routine.

Spectral Search Methods. With this software tool, one can search an unknown spectrum in various ways. Exact full spectrum search, regional search, and single peak search are implemented in this system.

(1) Exact full spectrum search: search for a spectrum exactly matched to a query spectrum. In the pre-search step, only the ''='' comparison operator and no wildcard character is used in the condition clause. One example query string is: IndexString = ''000Y00000000E0''.
(2) Regional search: search for a partial match spectra. In the pre-search step, only related region(s) is (are) subtracted for searching matched patterns with the ''='' or ''LIKE'' key words in the index table, and other regions are excluded. Multiple regions can be searched at the same time and the result can be combined logically. This search allows one to choose which regions of a spectrum to search, so that spectra of compound mixtures can be identified without the need for complex data manipulation. One can define the regions to exclude standard solvents and common interferents.
(3) Peak search: search for one or more peaks. There are two methods that can be used to search for single or multiple peaks: search in a peak table or an index table. A search in an index table can be performed easily: no matter whether a single peak or multiple peaks are being searched, only query peak regions are extracted to be compared with reference spectra in corresponding regions of the index table. Searching peaks in a peak

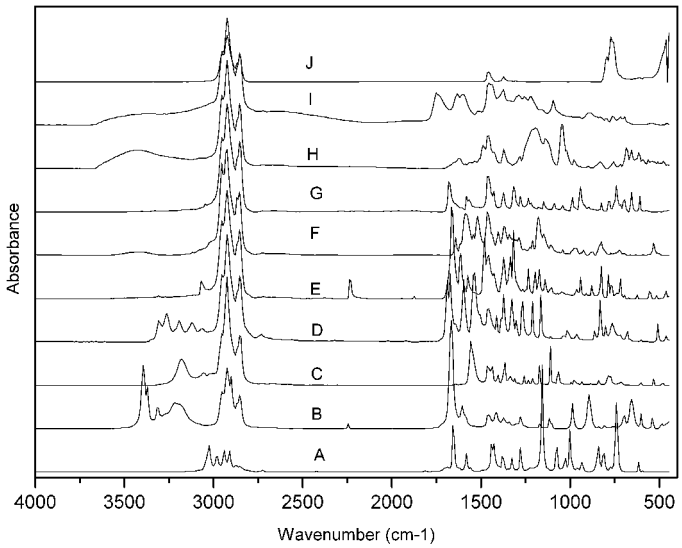


FIG. 3. The test spectra: (***A***) 2,7-dimethyloxepin, (***B***) omega-thiocaprolactam, (***C***) 2-amino-2-cyanoacetamide, (***D***) 4-acetamidobenzaldehyde, (***E***) 3-cyano-6-methylchromone, (***F***) trans-4-(4-(dibutylamino)styryl)-1-methylpyridinium iodide, (***G***) di-2-pyridyl ketone, (***H***) new coccine, (***I***) fluorexon, and (***J***) sodium iodate.

table is direct but a little difficult. A demonstration SQL sentence for searching two peaks in the 3300–3400 and 2800–2900 cm^{-1} regions is shown below. It is difficult to construct a predefined query string for multiple peak searching because the search engine doesn't know how many search peaks there are. Consequently, it needs to be constructed dynamically. Besides this limitation, it needs the key word ''distinct'' to eliminate the duplicate hits that will slow down the search speed, especially when many peaks are searched at the same time. There are no similar defects using the first method. The first method is stricter, which may result in the loss of some hits, but the second method can include intensity information in the query sentence.

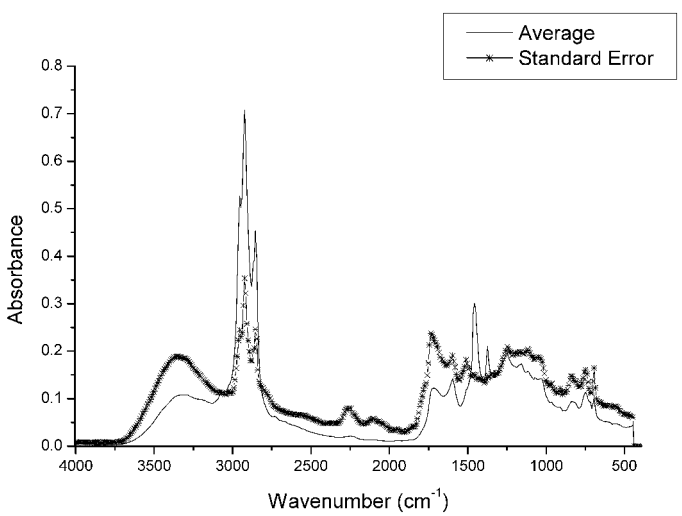


FIG. 4. Generic spectrum of the database.

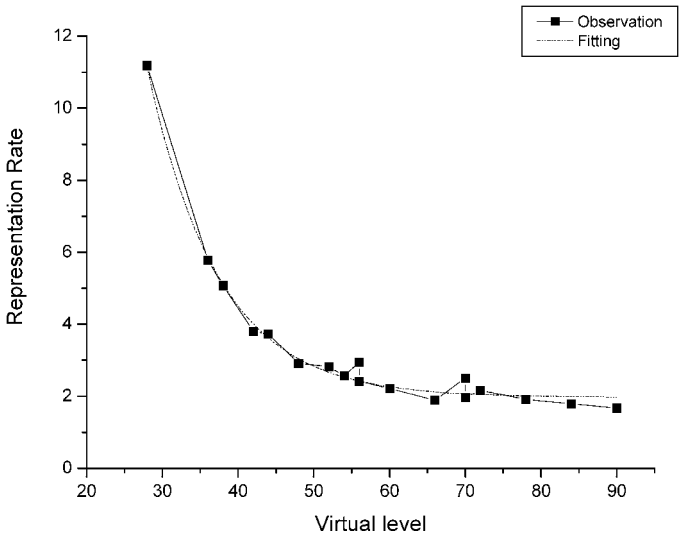


FIG. 5. The relationship between the virtual level and the representation rate.

```
SELECT DISTINCT(SpectrumID) AS SpecID
FROM PeakTable
WHERE SpectrumID IN (SELECT SpectrumID
FROM PeakTable WHERE (((PeakPos)>3300
AND (PeakPos)<3400))) AND SpectrumID IN
(SELECT SpectrumID FROM PeakTable
WHERE ((PeakPos)>2800 AND (PeakPos)
<2900))
```

General Steps to Search a Spectrum in a Spectral Database. The search of an unknown spectrum is performed in the following steps:

Step 1: A query spectrum is submitted to the database.

Step 2: The submitted spectrum is analyzed to construct a few query patterns in the same manner as creating the index string and to construct a canonical SQL query sentence.

Step 3: The database looks up the spectra in the index table to determine which rows in the ta-

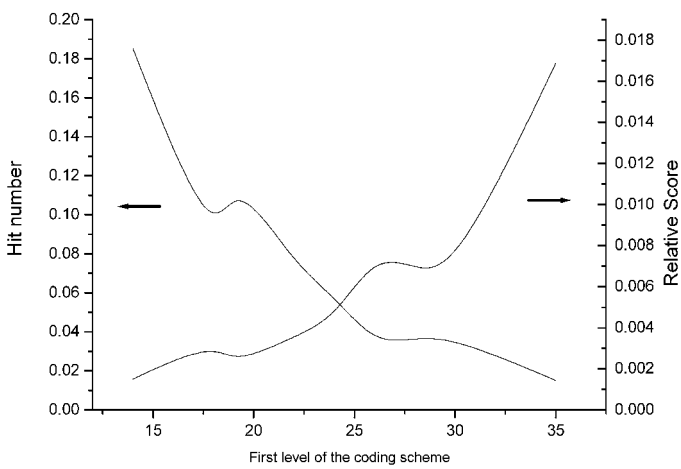


FIG. 6. The relationship between the first level of the coding scheme, hit number, and relative score. The relative score is the average score divided by the average hit number.

TABLE I. Average hits and scores of different code schemes for test compounds. (Second level of code scheme is 2.)

First level of code scheme	14	18	22	26	28	30	35	Average
Average hit number	3416.20	1486.20	1379.50	621.40	679.20	679.20	278.70	1318.05
Average score	0.51	0.49	0.49	0.47	0.47	0.45	0.47	0.48

ble are to be retrieved. All reference spectra matched with the query spectrum are selected. If the database returns nothing, the search routine stops; otherwise, it continues to Step 4.

Step 4: A hit list is ranked by Hit Quality Indexes (HQIs), which are calculated from the errors arising from a full spectrum comparison of the pre-search results in Step 3, in ascending order. A list of top scoring compounds is presented in the output.

For spectra like IR, UV/VIS/NIR, and Raman data, there are several possible algorithms to determine similarity between the query spectrum and library spectra in the full spectrum mode.[45,46] The most popular routines, which are point-by-point comparisons, used to compare two spectra are: Correlation, First Derivative Correlation, Euclidean Distance, Absolute Value, First Derivative Absolute Value, Least Squares, and First Derivative Least Squares algorithms. They are all time-consuming routines. In the Correlation algorithm, both the unknown and the library data are centered about their respective means before the vector dot products are calculated. It is independent of the spectral range of the search spectrum and the normalization of the spectra. In almost all cases, the Correlation algorithm will provide better or equal hit quality information when compared to other searching methods.[3,47] In our system, the Correlation algorithm is used for calculating the HQIs. It has a value of 1 for perfectly matched spectra.

$$HQI = \frac{(Lib_m \cdot Unkn_m)^2}{(Lib_m \cdot Lib_m)(Unkn_m \cdot Unkn_m)} \quad (5)$$

where the vectors are defined as:

$$Lib_m = Lib - \frac{\sum_{i=1}^{n} Lib_i}{n} \quad (6)$$

$$Unkn_m = Unkn - \frac{\sum_{i=1}^{n} Unkn_i}{n} \quad (7)$$

where *Lib* is the library entry being searched and *Unkn* is the unknown spectrum.

EXPERIMENT

Spectral Database. The spectra are collected from Aldrich FT-IR Collection Edition, which is a library of the compounds offered by Sigma-Aldrich in the Aldrich Catalog/Handbook of Fine Chemicals for general laboratory work. The spectra in this library were published in the Aldrich Library of FT-IR Spectra, volumes 1, 2, and 3 (copyright 1998, Nicolet Instrument Corporation). The wavenumber region is from 4000 to 440 cm^{-1}, and there are 496 points in one spectrum, which means the corresponding resolution is 7.714 cm^{-1}.

Test Spectra Set. In order to evaluate data preprocessing techniques, a test data set is composed with ten random spectra collected from the database: (1) 2,7-dimethyloxepin, (2) omega-thiocaprolactam, (3) 2-amino-2-cyanoacetamide, (4) 4-acetamidobenzaldehyde, (5) 3-cyano-6-methylchromone, (6) trans-4-(4-(dibutylamino)styryl)-1-methylpyridinium iodide, (7) di-2-pyridyl ketone, (8) new coccine, (9) fluorexon, and (10) sodium iodate. Their spectra are shown in Fig. 3.

RESULTS AND DISCUSSION

Number of Sections and Subsections. The code method is a kind of compression method. Representing a continuous spectrum by a code string may result in some loss of information due to the discrete nature of the encoding. Losing some spectral detail is definite and inevitable. We can get a statistical snapshot of the data distribution of our library from the averages and standard

TABLE II. Hit number of the search with different window widths.

Width[a]	5	25	50	75	100
1	109	411	411	78	477
2	18	170	170	550	2571
3	29	137	463	119	679
4	78	107	441	2388	4012
5	8	102	14	14	27
6	447	1479	1586	289	2074
7	686	1182	1656	738	2063
8	783	1230	1230	832	1944
9	456	637	1777	1179	1849
10	954	1704	1704	954	1810
Average	356.8	715.9	945.2	714.1	1750.6

[a] (1) 2,7-Dimethyloxepin, (2) omega-thiocaprolactam, (3) 2-amino-2-cyanocetamide, (4) 4-acetamidobenzaldehyde, (5) 3-cyano-6-methylchromone, (6) trans-4-(4-(dibutylamino)styryl)-1-methylpyridinium iodide, (7) di-2-pyridyl ketone, (8) new coccine, (9) fluorexon, and (10) sodium iodate.

TABLE III. Scores of the search with different window widths.

Width[a]	5	25	50	75	100
1	0.13	0.17	0.17	0.13	0.17
2	0.66	0.69	0.69	0.79	0.81
3	0.49	0.40	0.53	0.42	0.55
4	0.54	0.54	0.67	0.72	0.73
5	0.55	0.36	0.58	0.42	0.45
6	0.84	0.86	0.86	0.79	0.86
7	0.84	0.87	0.89	0.86	0.89
8	0.83	0.85	0.85	0.83	0.87
9	0.74	0.76	0.80	0.79	0.80
10	0.48	0.50	0.50	0.48	0.50
Average	0.61	0.60	0.65	0.62	0.66

[a] Compound names are the same as in Table II.

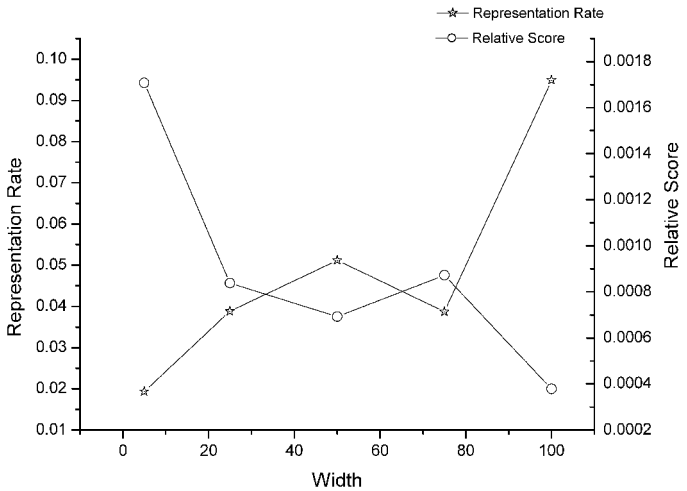


FIG. 7. The relationship between the window width, representation rate, and relative score. Representation rate is the hit number divided by the total number of spectra in the database. The relative score is the average score divided by the average hit number.

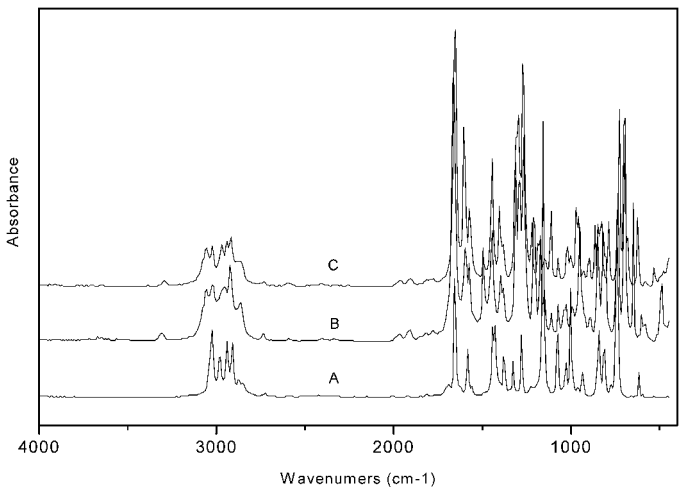


FIG. 8. Comparison of spectra of (***A***) 2,7-dimethyloxepin, (***B***) 2,5-dimethylbenzophenone, and (***C***) 3,4-dimethylbenzophenone.

errors of all points of the spectra in the library, which is shown in Fig. 4. If the average is bigger than the standard error, the point is almost the same one in most spectra. From Fig. 4 one can find that the standard errors are higher than the averages in most of the ranges of the generic spectrum. The distribution of peaks is not even. The number of peaks per spectrum and the density of peaks are different spectrum to spectrum. Generally, to present a good spectrum in code string, one should allocate more space for spectra with high peak density. In general, one cannot find a code scheme that is suitable for every spectrum and not lose any band information. To investigate the influences of the first and the second level of the coding scheme, we carried out systematic tests. For convenience, we defined two items—virtual level and representation rate—for the comparison of the effect of the level of the coding scheme. Virtual level is the product of the first and second levels of the coding scheme. Representation rate is the number of distinct rows divided by the total number of spectra in the database. The results are shown in Fig. 5. The representation rate decreases when the virtual level increases. The regulation can be described by the following formula: $Representation\ rate = 1.96 + 184.31 \times \exp(-virtual\_level/9.33)$. That is to say, the first and second levels of the code scheme have similar influences on the representation rate.

Most of the compounds in the library are alkanes, which have similar strong bands in the functional group region. For example, in our 10 test-set compounds, almost half of the test set has very similar bands in the functional group region. These spectra represent the major part of the spectra in the library. As we emphasized before, there is no code system suited to all spectra. Therefore, to balance both the major and minor parts of the spectra in the database we must find a tolerable compression ratio to trade off between the distinct rate and the number of hits. The score of the results depends on the hit number of the search. On the other hand, it is clear that if the pre-search result set is too big, the index will have little meaning. The higher the distinction rate is, the higher multiplicity is, and the fewer the number of hits. One can improve fetch performance by creating appropriate indexes and properly tuning the query syntax to leverage the spectral search. For investigating the balance point of hit number and score, we used the relative score (score divided by hit number). The result is shown in Fig. 6. Unfortunately, we cannot find a platform for either hit number or relative score. They decrease or increase monotonously. From Table I, we find that the score is the highest, 0.51, when the first level is 14. At the same time, it has the biggest hit number. The average hit number is 1318.05, and the biggest one is about 2.5 times the average value. Compared with the hit number, the variation of the scores is small: the biggest is 0.51 and the smallest is 0.45. Both maximum and minimum values are close to the average. Because the score is above the average and the hit number is 278.70, and that is enough for the general comparison of spectra, we chose 35 as a suitable number for the first level of the coding scheme.

Effects of the Window Width. From the data listed in Tables II and III, we can see that the number of hits grows with the extension of the width of the window. When the hit number is bigger, the score is higher as well. The improvement of the score is not proportional to the improvement of the hit number. The hit numbers vary from 356.8 to 1750.6 and the scores vary from 0.60 to 0.66. The hit number changes more dramatically than the score. To show the result clearly, we have drawn the representation rates and relative scores on the same figure (Fig. 7). In Fig. 7, there is a platform where the relative scores and representation rates change a little in the width range from 25 to 75. For finding better results in a bigger candidate set, the search routine needs more time and resources.

Exact Full Spectrum Search. The results of the exact full spectrum search of 10 test spectra are listed below. Seven of them have less than three candidates and three of them are found directly in the library. Only for sodium iodate have 734, more than 100 hits, been found for

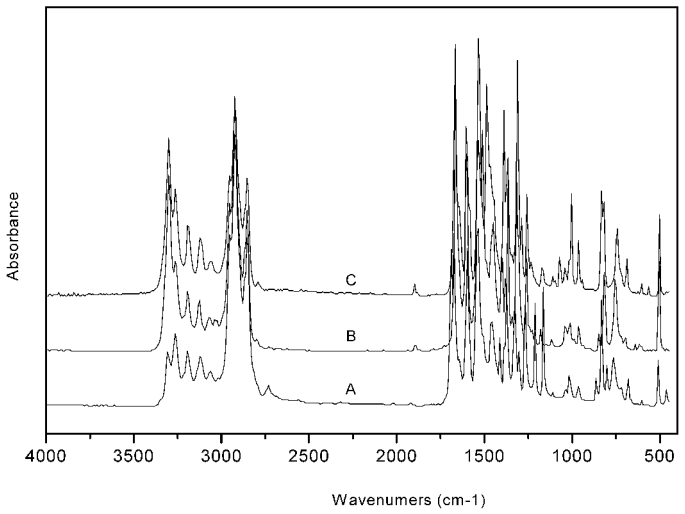
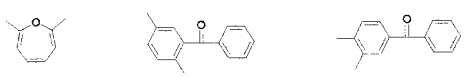


FIG. 9. Comparison of spectra of (***A***) 4-acetamidobenzaldehyde, (***B***) 4′-methylacetanilide, and (***C***) 4'-bromoacetanilide.

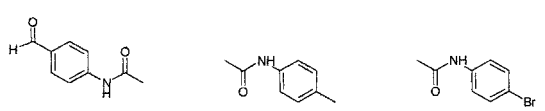
matched spectra in the functional group region. That is to say, for most samples, the discrimination ability is high. In Figs. 8 and 9, we show a comparison of the spectra for two series of compounds:

SERIES 1:



2,7-dimethyloxepin 2,5-dimethylbenzophenone 3,4-dimethylbenzophenone

SERIES 2:



4-acetamidobenzaldehyde 4′-methylacetanilide 4′-bromoacetanilide

The results show that although the structures are different, the spectra are obviously very similar (Table IV).

Effect of the Shifted Query Strings. To test the performances of the shifted strings, the top 100 hits of different search strings are saved and the average is calculated for them. If the hit number is less than 100, the average is calculated on the actual number of hits. The hit numbers of the results of the three kinds of query strings and the total distinct results are shown in Table V. In most cases, two shifted strings create strong supplements to the fuzzy query string. The scores have been improved. The qualities of the shifted strings are close to the results of the fuzzy strings.

Similarity Search. One of the most interesting philosophical points of infrared spectral searching revolves around the assumption that similar infrared spectra correspond to organic compounds with similar molecular structures. This assumption becomes particularly important when the unknown spectrum is not found in the database being searched. When this occurs, the best matches may or may not correspond to compounds with very similar structures. The notion of similar molecular structure is completely dependent on the specific application and objectives of the user. In one case, the fact that the best matches are all aromatic esters may be sufficient, while in another case, the exact substitution pattern is the key piece of information. While a search algorithm can be optimized for a particular application, this may require a specialized reference library specifically designed for the application.

By using the method presented in this report, one can get more similar spectra from a large-volume database efficiently. This will benefit the structure interpretation because similar structures have similar spectra. The comparison of scores is shown in Table VI. From the results we can find that a one-by-one search is always better than this method. In some cases, the results of this system are close to that of a one-by-one search. From the data in Table VI, the average compression ratio is 0.039, about 714.10 spectra in one search. The comparison between the results of this method and a one-by-one search can be illustrated by the spectrum of 2-amino-2-cyanoacetamide. The scores of the top 5 hits for this method and a one-by-one search are 0.66 and 0.68 respectively. The size of the candidate set for this method is 119, only 0.67% of the total spectra in the library. The result is very similar to the one-by-one search of the total 18 454

TABLE IV. Exact full spectrum search results.

Hit count	1	2	3	20–35	Other
Number of samples	3	1	3	2 (25, 31)	1 (735)

TABLE V. Results of the different query sentences.

	Hits			Score		
Sample[a]	3 Strings	Fuzzy	Shifted	3 Strings	Fuzzy	Shifted
1	78	44	78	0.13	0.14	0.13
2	550	440	516	0.79	0.79	0.79
3	119	33	86	0.42	0.41	0.36
4	2388	1452	2388	0.72	0.70	0.72
5	14	9	11	0.42	0.54	0.36
6	289	278	182	0.79	0.79	0.74
7	738	402	361	0.86	0.82	0.79
8	832	324	510	0.83	0.79	0.75
9	1179	205	1135	0.79	0.67	0.79
10	954	886	954	0.48	0.48	0.48

[a] Compound names are the same as in Table II.

Table VI. Comparison of top 5 similarity search results of this system and a one-by-one search.

Sample[a]	This system	One-by-one method
1	0.21	0.29
2	0.85	0.86
3	0.66	0.68
4	0.77	0.78
5	0.64	0.84
6	0.89	0.93
7	0.92	0.93
8	0.95	0.95
9	0.91	0.92
10	0.62	0.63
Average	0.74	0.78

[a] Compound names are the same as in Table II.

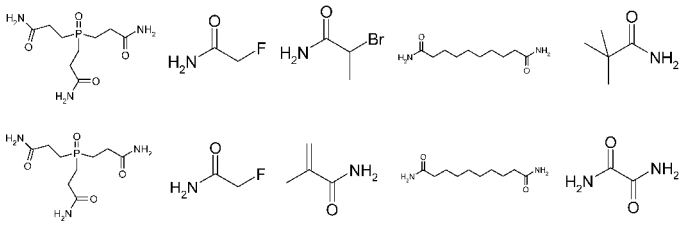


FIG. 10. The top 5 similar structures found by the one-by-one method (above) and the method in this report (below).

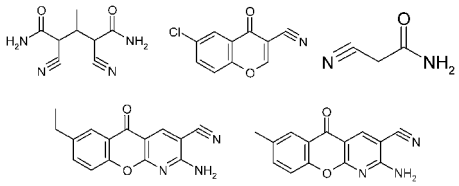


FIG. 11. Result structures of the region spectrum search of 2-amino-2-cyanoacetamide.

spectra. All the search results have similar structures with the query compound (Fig. 10).

Regional Spectrum Search. For the sample spectrum of 2-amino-2-cyanoacetamide, most of the bands in the functional group region have corresponding bands in similar spectra found by the similarity search. But the peak at around 2247 cm^{-1} is missing from all the top 5 highest score spectra because it is too weak. The correlation algorithm overlooks the contribution of this weak band. The region spectrum search can help to investigate spectra that have similar bands in special regions. For this example, one can find a total of 427 candidates by region search. The average score of the top 5 in the functional group region is 0.56. They are 2,4-dicyano-3-methylglutaramide, 6-chloro-3-cyanochromone, 2-cyanoacetamide, 2-amino-7-ethyl-5-oxo-5h-(1)benzopyrano (2,3-b)pyridine-3-carbonitrile, and 2-amino-7-methyl-5-oxo-5h-(1)benzopyrano (2,3-b)pyridine-3-carbonitrile. The new common functional group is C≡N, which is a common functional group in the previous search results. By this analysis, one can say that the query example 2-amino-2-cyanoacetamide is a compound that has a cyanophoric C≡N functional group (Fig. 11).

CONCLUSION

The results of this research indirectly indicate that the lack of intensity information doesn't influence the result of spectrum searching. The fuzzy query string and the two strings shifted in opposite directions greatly reduce the probability of losing some candidates by using the SQL search. The width window compensates for the defects created by the transformation from a continuous spectrum to a discrete code string. Also, SQL library searching can lead to an order of magnitude reduction in the search space by using a coding index. Examples proved that using the indexes enhanced the library's selectivity dramatically, as the size of candidate sets decreased from 18 454 to less than 1000 and in some cases less than 100. Such a saving can improve the search speed and the chance of getting real, relevant results from large library search systems. It has also been demonstrated in this report that the code string can be used for similarity by taking advantage of RDBMS and SQL search methods. One thing that should be mentioned is that there is no one method that is suitable for all search aims for all kinds of spectra, but a method may run better for most of the computerized spectral identifications.

Generally speaking, storing and managing the spectra library by a relational database system will make the management of the data easier and will have more potential benefits. It is possible to store all chemical information, i.e., the molecular structures, kinds of physical and chemical properties, spectra, etc., in a comprehensive relational database. The mature and powerful SQL search paths make it is possible to create a flexible spectral search to satisfy different demands with high efficiency. Also, it is easy to publish the data in a normal commercial relational database on the Web and to meet the needs of web queries.

1. B. G. Derendyaev, L. I. Makarov, T. F. Bogdanova, and V. N. Piottukh-Peletskii, J. Struct. Chem. **42,** 271 (2001).
2. M. E. Munk, J. Chem. Inf. Comput. Sci. **38,** 997 (1998).
3. P. N. Penchev and K. Varmuza, Comput. Chem. **25,** 231 (2001).
4. G. W. Somsen, I. Jagt, C. Gooijer, N. H. Velthorst, and U. A. Brinkman, J. Chromatogr., A **756,** 145 (1996).
5. W. A. Warr, Chemom. Intell. Lab. Syst. **10,** 279 (1991).
6. Bio-Rad Laboratories, Inc: http://www.bio-rad.com.
7. Sadtler Research Laboratories, *Sadtler Standard Infrared Grating Spectra* (Division of Bio-Rad Laboratories, Philadelphia, 1980).
8. C. J. Pouchert, *Aldrich Library of Infrared Spectra* (Aldrich Chemical Company, Milwaukee, WI, 1981), 3rd ed.
9. C. J. Pouchert, *Aldrich Library of FT-IR Spectra* (Aldrich Chemical Company, Milwaukee, WI, 1985), 1st ed.
10. R. J. Kelley, *The Sigma Library of FT-IR Spectra* (Sigma Chemical Co., 1986).
11. D. O. Hummel, *Atlas of Polymer and Plastics Analysis* (VCH, New York, 1985).
12. American Society for Testing and Materials, *Alphabetical List of Compound Names, Formulae, and References to Published Infrared Spectra: an Index to 92,000 Published Infrared Spectra* (ASTM, Philadelphia, 1969).
13. A. N. Davies, Spectroscopy Europe **12,** 30 (2000).
14. Chemical Concepts: http://www.chemicalconcepts.com.
15. SpecInfo database: http://www.library.qmw.ac.uk/sci/cds.htm.
16. NIST Chemistry WebBook: http://webbook.nist.gov /chemistry/.
17. Integrated Spectral Data Base System for Organic Compounds (SDBS): http://www.aist.go.jp/RIODB/SDBS/intro-en.html.
18. M. I. Podgornaya and B. G. Derendyaev, ''Databases on IR spectroscopy of organic compounds'', Nauchno-Tekh. Inf., Ser. 2, No. 9, 1 (1992).
19. M. Meyer, I. Weber, R. Sieler, and H. Hobert, Jena Rev. **35,** 16 (1990).
20. C. I. Gerhausser and K. Kovar, Appl. Spectrosc. **51,** 1504 (1997).
21. K. Tanabe, K. Hayamizu, and S. Ono, Anal. Sci. **7,** 711 (1991).
22. D. F. Averill, K. C. Baird, L. L. Hopkins, and M. J. Yerkes, J. Chem. Inf. Comput. Sci. **30,** 133 (1990).
23. T. D. Jarvis and J. H. Kalivas, Anal. Chim. Acta **272,** 53 (1993).
24. S. K. Smith, J. Cobleigh, and V. Svetnik, J. Chem. Inf. Comput. Sci. **41,** 1463 (2001).
25. S. V. Trepalin and A. V. Yarkov, J. Chem. Inf. Comput. Sci. **41,** 100 (2001).
26. S. R. Lowry, D. A. Huppler, and C. R. Anderson, J. Chem. Inf. Comput. Sci. **25,** 235 (1985).
27. C. Cai, P. de Harrington, and D. M. Davis, Anal. Chem. **69,** 4249 (1997).
28. R. B. Lam, S. J. Foulk, and T. L. Isenhour, Anal. Chem. **53,** 1679 (1981).
29. A. K. Leung, F. Chau, J. Gao, and T. Shih, Chemom. Intell. Lab. Syst. **43,** 69 (1998).

30. B. Walczak and J. P. Radomski, Data Handl. Sci. Technol. **22,** 291 (2000).
31. S. C. Lo and C. W. Brown, Appl. Spectrosc. **45,** 1628, (1991).
32. B. K. Alsberg, J. Chemom. **7,** 177 (1993).
33. E. K. Kemsley, Chemom. Intell. Lab. Syst. **33,** 47 (1996).
34. M. Hideyuki and Y. Mototsugu, J. Chem. Inf. Comput. Sci. **36,** 294 (1996).
35. Y. P. Drobyshev, R. S. Nigmatullin, V. I. Lobanov, I. K. Korobeinicheva, V. S. Bochkarev, and V. A. Koptyug, Vestn. Akad. Nauk SSSR **40,** 75 (1970).
36. E. C. Penski, D. A. Padowski, and J. B. Bouck, Anal. Chem. **46,** 955 (1974).
37. R. C. Fox, Anal. Chem. **48,** 717 (1976).
38. F. H. Heite, P. F. Dupuis, H. A. Van't Klooster, and A. Dijkstra, Anal. Chim. Acta **103,** 313 (1978).
39. S. L. Ellison and S. L. Gregory, Anal. Chim. Acta **370,** 181 (1998).
40. D. A. Skoog and D. M. West, *Principles of Instrumental Analysis* (Saunders, Philadelphia, 1980), 2nd ed., p. 232.
41. D. H. Anderson and G. L. Covert, Anal. Chem. **39,** 1288 (1967).
42. W. S. William, *The Sadtler Handbook of Infrared Spectra* (Sadtler Research Laboratories, Philadelphia, 1978).
43. K. Tanabe, T. Tamura, J. Hiraishi, and S. Saeki, Anal. Chim. Acta **112,** 211 (1979).
44. O. Lau, P. Hon, and B. Tao, Vib. Spectrosc. **23,** 23 (2000).
45. P. N. Penchev, A. N. Sohou, and G. N. Andreev, Spectrosc. Lett. **29,** 1513 (1996).
46. P. N. Penchev, N. T. Kotchev, and G. N. Andreev, Dokl. Bulg. Akad. Nauk. **51,** 67 (1998).
47. D. L. Massart, B. G. M. Vandeginste, L. C. M. Buydens, S. de Jong, P. J. Lewi, and J. Smeyers-Verbeke, *Handbook of Chemometrics and Qualimetrics: Part A* (Elsevier, Amsterdam, 1997).